General

The string method `index`

The string method `index` may be used to determine the position of the leftmost occurrence of a character in a strings.

```
>>> string = 'mississippi'
>>> character = 'i'
>>> string.index(character)
1
```

This code fragment finds the leftmost occurrence of the letter `i` at position 1 in the string `mississippi`. Note that Python indexes the positions of the characters in a string starting from 0, not starting from 1.

It should be noted that this methode might be inefficient in case the position of the character can be derived directly. In order to find the position of a character that first occurs at position p, the index method must traverse the first $p-1$ positions of the string. For a character that does not occur in the string, the string method index must even travers all characters in the string before it can decide that the character was not observed.

Remove leading and/or trailing whitespace

In Python you can use the string method `strip` to remove leading and trailing whitespace (spaces, tabs and newlines). In case you only want to remove leading whitespace, you can use the string method `lstrip`. In case you only want to remove trailing whitespace, you can use the string method `rstrip`. You can also pass an argument to these string methods, that indicates which leading and/or trailing characters have to be removed. For more details about these string methods, we refer to The Python Standard Library.

```
>>> text = '  This is a text  '
>>> text.strip()
'This is a text'
>>> text.lstrip()
'This is a text  '
>>> text.rstrip()
'  This is a text'
>>> text2 = '===This is a text==='
>>> text2.lstrip('=')
'This is a text==='
```

Atbash

Convert letters to their ordinal value

In Python, each character has a corresponding ordinal value (an integer). For example, the question mark (`?`) has ordinal value 63. The value can be easily obtained using the built-in function `ord`.

```
>>> ord('?')
63
```

The interesting thing about these ordinal values, is that successive letters in the alphabet have successive numerical values. This holds for both uppercase and lowercase letters. If we know that the letter `a` has ordinal value 97, it immediately follows that the letter `b` must have ordinal value 98. With this knowledge, we can easily determine the position of a letter in the alphabet.

```
>>> letter = 'd'
>>> ord(letter)
100
```

```
>>> ord('a')
97
>>> ord(letter) - ord('a') + 1   # d is the 4th letter of the alphabet
4
>>> ord('z') - ord('a') + 1      # z is the 26st letter of the alphabet
26
```

String methods `islower()` and `isupper()`

The string method `islower` checks if *all* characters of a string are lowercase letters. The string method `isupper` checks if *all* characters of a string are uppercase letters.

```
>>> 'ABCD'.lower()
False
>>> 'ABCD'.upper()
True
>>> 'abcd'.lower()
True
>>> 'abcd'.upper()
False
>>> 'AbCd'.lower()
False
>>> 'AbCd'.upper()
False
```

Wepe speapeak p

Specific information

We'll provide a possible strategy you could use in this exercises. First you iterate over all positions in a line and you'll keep track of two variables. The first variable should in the end contain the decoded text, the second variable contains the current vowel group. When you come across the letter 'p' while iterating en the vowel group is not empty, this vowel group can be processed and you may jump a few positions ahead in the encoded message. See the attached figure for a schematic representation.

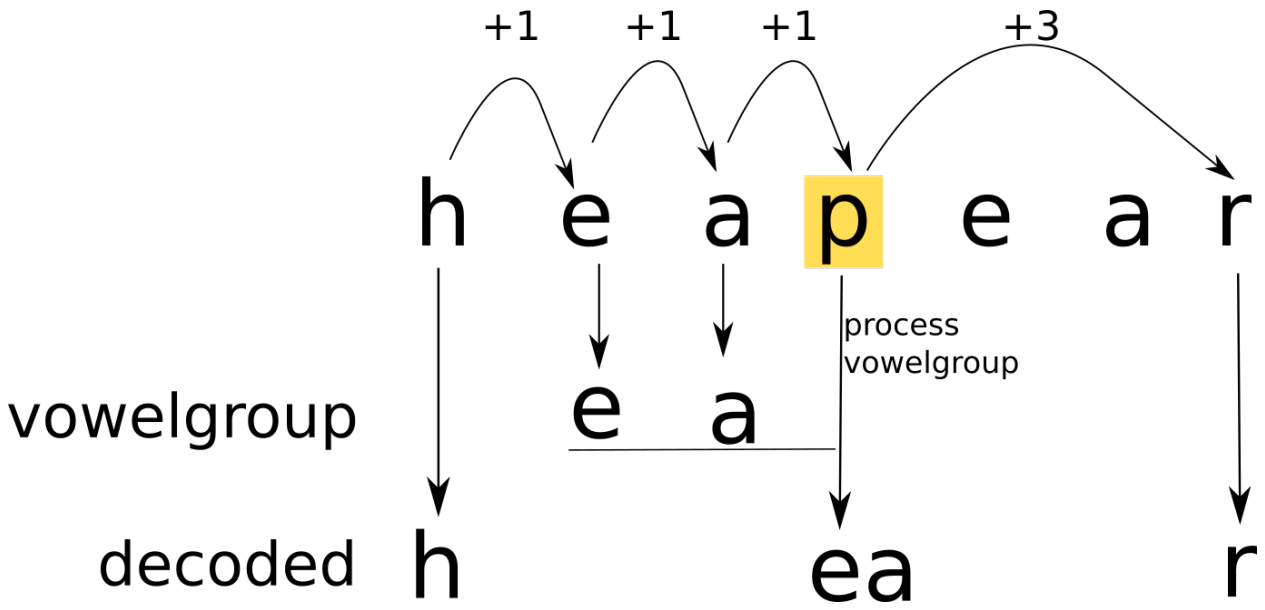

Figure 1: Wepe speapeak p